OpenStreetMaps Project- Mapping Trenton, Princeton & Plainsboro City Area, New Jersey

Udacity Data Science Nanodegree course- Final Project

Data Wrangling with Mongo DB

By: Mubina Arastu

MArastu@wgu.edu

Summary

This project details my analysis and cleanup of the OpenStreetMap.org map data for the majority of Trenton, Princeton & Plainsboro cities of New Jersey. I focused on street addresses and amenities listed within the map.I choose these cities because I live and work in these areas.

Section 1: Problems Encountered in the Map

I have used sample OSM data 9.5 MB (9528 kb) and the original size of OSM file (150 MB) for this project. Both the compressed XMLs are provided for the reviewer. This writeup is based on original XML data.

This writeup list out few problems that were encountered in the dataset, and gives an high level overview of the data. Exploratory data analysis, wrangling and cleaning of this project can be seen in python notebook.

Problems With the Dataset

Initialy I faced difficulty in downloading a reasonable size dataset. After few attempts, I finally downloaded sample OSM file that includes Trenton area of New Jersey. As per my exploratory analysis of the dataset, I found few issues that need to be cleaned up:

1. Over abbreviated city names like (“Twp”,"twp) to clean up.
2. Wrong street type like 'Nassau Parl Blvd" to clean up
3. Brand names doesn't have a 'type' tag. "Wikipedia" tag need to be updated as "type".
4. Amenities like shops have some wierd data like 'doityourself" to clean up.

In [107]:
```
# Using update_one function to update 'twp' to Township
city=db.TrentPrincemaps
city.update_many({'address.city': 'Hopewell twp'}, {'$set': {'address.city':
'Hopewell Township'}})
city.update_many({'address.city': 'Ewing twp'}, {'$set': {'address.city': 'Ewi
ng Township'}})
city.update_many({'address.city': 'Princeton twp'}, {'$set': {'address.city':
'Princeton Township'}})
city.update_many({'address.city': 'West windsor twp'}, {'$set': {'address.cit
y': 'West Windsor Township'}})
city.update_many({'address.city': 'Lawrence Twp'}, {'$set': {'address.city':
'Lawrence Township'}})
```

Out[107]:
```
<pymongo.results.UpdateResult at 0x19f0cd31740>
```

In [108]:
```
# Using update_many function to clean street error
street=db.TrentPrincemaps
street.update_many({'address.street': 'Nassau Parl Blvd'}, {'$set': {'address.
street': 'Nassau Park Boulevard'}})
street.update_many({'address.street': 'Nassau Park Blvd'}, {'$set': {'address.
street': 'Nassau Park Boulevard'}})
```

Out[108]:
```
<pymongo.results.UpdateResult at 0x19f0bd37a80>
```

Section 2: Data Overview

Find Documents

In [110]:
```
db.TrentPrincemaps.count_documents({})
```

Out[110]:
```
818060
```

Find Nodes

In [111]:
```
#Find Nodes
nodes=db.TrentPrincemaps.count_documents({'type':'node'})
pprint.pprint("count of nodes")
pprint.pprint(nodes)
```

```
'count of nodes'
732944
```

Find Ways

In [112]:
```
#Find Ways
ways=db.TrentPrincemaps.count_documents({'type':'way'})
pprint.pprint("count of ways")
pprint.pprint(ways)
```

```
'count of ways'
85116
```

Find unique Users

In [113]:
```
u=len(db.TrentPrincemaps.distinct('created.user'))
pprint.pprint("Number of Unique Users")
pprint.pprint(u)
```

```
'Number of Unique Users'
1439
```

Top 5 Users

In [114]:
```
topusers  = db.TrentPrincemaps.aggregate([
    {"$group" :
        {"_id" : "$created.user",
         "count" : {"$sum" : 1}}},
          { '$sort' : { 'count' : -1 } }, { '$limit' : 5 }
                              ])
for k in topusers:
    print(k)
```

```
{'_id': 'RM23', 'count': 91641}
{'_id': 'woodpeck_fixbot', 'count': 70444}
{'_id': 'NJDataUploads', 'count': 46779}
{'_id': 'OceanVortex', 'count': 31600}
{'_id': 'Utible', 'count': 29535}
```

Find Amenities & Highways

In [115]:
```
#Find Service Highways
s=db.TrentPrincemaps.count_documents({"highway": "service"})
pprint.pprint("count of services")
pprint.pprint(s)

#Find Residential Highways
r=db.TrentPrincemaps.count_documents({"highway": "residential"})
pprint.pprint("count of residential areas")
pprint.pprint(r)

#Find number of parking sites
p=db.TrentPrincemaps.count_documents({"amenity": "parking"})
pprint.pprint("count of parking areas")
pprint.pprint(p)
```

```
'count of services'
1656
'count of residential areas'
237
'count of parking areas'
216
```

List number of shops

In [119]:
```
## Aggregate function is used to show shops based on counts sorted in ascendin
g order
shops  = db.TrentPrincemaps.aggregate([
    {"$group" :
        {"_id" : "$shop",
         "count" : {"$sum" : 1}}},
          { '$sort' : { 'count' : 1 } }, { '$limit' : 5 }
                             ])
for sh in shops:
    print(sh)
```

```
{'_id': 'doityourself', 'count': 1}
{'_id': 'tyres', 'count': 3}
{'_id': 'cosmetics', 'count': 3}
{'_id': 'party', 'count': 3}
{'_id': 'baby_goods', 'count': 3}
```

From above data one wierd name 'doityourself' is seen as a shop category. Will change doityourself to 'home_improvement'.

In [120]:
```
# Using update_one function to update 'doityourself' to home_improvement
coll=db.TrentPrincemaps
updateResult = coll.update_one({'shop': 'doityourself'}, {'$set': {'shop': 'ho
me_improvement'}})
```

check if 'doityourself is updated to home_improvement with 1 count'

In [121]:
```
## Using aggregate function with limit to show only 30 top shops sorted in des
cending order.

updatedshops  = db.TrentPrincemaps.aggregate([
    {"$group" :
        {"_id" : "$shop",
         "count" : {"$sum" : 1}}},
          { '$sort' : { 'count' : 1 } }, { '$limit' : 1}
                              ])
for upsh in updatedshops:
    print(upsh)
```

```
{'_id': 'home_improvement', 'count': 1}
```

Section3: Additional Exploration

Number of Offices

In [122]:
```
topoffices  = db.TrentPrincemaps.aggregate([
    {"$group" :
        {"_id" : "$office",
         "count" : {"$count" : {}}}},

                              ])
for k in topoffices:
    print(k)
```

```
{'_id': None, 'count': 818042}
{'_id': 'energy_supplier', 'count': 3}
{'_id': 'government', 'count': 9}
{'_id': 'company', 'count': 6}
```

List out Top 5 Users Posts

In [75]:
```
topusers  = db.TrentPrincemaps.aggregate([
    {"$group" :
        {"_id" : "$created.user",
         "count" : {"$count" : {}}}},
     { '$sort' : { 'count' : -1 } }, { '$limit' : 3 }
            ])
for k in topusers:
    print(k)
```

```
{'_id': 'iskra32', 'count': 13680}
{'_id': 'OceanVortex', 'count': 8193}
{'_id': 'ogm17', 'count': 3102}
{'_id': 'Utible', 'count': 2892}
{'_id': 'Nikhil K', 'count': 2685}
```

List out users with only 1 Post

In [123]:
```
leastusers  = db.TrentPrincemaps.aggregate([
{"$group":{"_id":"$created.user", "count":{"$sum":1}}},
                       {"$group":{"_id":"$count", "num_users":{"$sum":1}}},
                       {"$sort":{"_id":1}}, {"$limit":1}])
for k in leastusers:
    print(k)
```

```
{'_id': 1, 'num_users': 163}
```

Section 3: Additional Ideas

Contributor statistics and gamification suggestion: The contributions of users seems incredibly skewed. Here are some user percentage statistics:

Top user contribution percentage (“iskra32”) - 26% Combined top 2 users' contribution (“iskra32”)” and “OceanVortex”) - 42% Combined Top 10 users contribution - 62.20% Users making up only 5% of posts (1/51079)

Looking into these user percentages, I believe “gamification” is a motivating factor for contribution. In the context of the OpenStreetMap, if users data are more accurately displayed then more people will take interest in submitting edits to the map data.

Section 4: Conclusion

After this review of the data it’s obvious that the Trenton city area is incomplete, though I have cleaned up well for the purpose of this project. However, the improvement may bring more issues if not implemented correctly:

1.Gamification may impact the quality of the data submitted from the contributors. We should keep in mind quality is more important then quantity.

2.If the difference between the highest score and other scores is too large, users may easily loose their interest. Therefore, we should implement the logic in such a way that as the scores rise high, it should become more difficult to increase.

With a robust data processor, it would possibly input more cleaner data to OpenStreetMap.org.